Noname manuscript No.
(will be inserted by the editor)

A Modular Approach to Learning Manipulation Strategies from Human Demonstration

Bidan Huang, Miao Li, Ravin Luis De Souza, Joanna J. Bryson, Aude Billard

the date of receipt and acceptance should be inserted later

Abstract Object manipulation is a challenging task for robotics, as the physics involved in object interaction is complex and hard to express analytically. Here we introduce a modular approach for learning a manipulation strategy from human demonstration. Firstly we record a human performing a task that requires an adaptive control strategy in different conditions, i.e. different task contexts. We then perform modular decomposition of the control strategy, using phases of the recorded actions to guide segmentation. Each module represents a part of the strategy, encoded as a pair of forward and inverse models. All modules contribute to the final control policy; their recommendations are integrated via a system of weighting based on their own estimated error in the current task context. We validate our approach by demonstrating it, both in a simulation for clarity, and on a real robot platform to demonstrate robustness and capacity to generalise. The robot task is opening bottle caps. We show that our approach can modularize an adaptive control strategy and generate appropriate motor commands for the robot to accomplish the complete task, even for novel bottles.

B. Huang (✉)
The Hamlyn Centre, Imperial College London, United Kingdom
E-mail: b.huang@imperial.ac.uk

J.J. Bryson
Intelligent Systems Group (IS), Computer Science Department, University of Bath, United Kingdom
E-mail: jjb@cs.bath.ac.uk

M.Li · R.L. de Souza · A. Billard
Learning Algorithms and Systems Laboratory (LASA), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
E-mail: miao.li@epfl.ch

R.L. de Souza
E-mail: ravin.desouza@epfl.ch

A. Billard
E-mail: aude.billard@epfl.ch

1 Introduction

With robots moving into human-centered environments such as households and offices, human-like motor skills are becoming increasingly desirable. In everyday life, object manipulation is one of the most commonly used manual skills. Object manipulation includes a large category of activities ranging from the simple pick-and-place task to complicated dexterous manipulation.

Here we provide a framework for learning a human object-manipulation skill and transferring it to a robot. Generally, manipulation tasks are very difficult, due to the complicated contact situations between the manipulator and the object, and the changing kinematic and dynamic contexts that result. Humans can perform these skilled tasks and adapt to changes in context without difficulty. At the heart of this skill is prediction (Flanagan et al, 2006). Studies from neuroscience suggest that humans develop internal models for motor control, which allow us to predict the future state of the environment. By comparing the predictive state with the actual sensory state, the internal models monitor the progression of tasks, and launch any corresponding motor correction and motor reaction required to adapt to anything unexpected.

Inspired by this concept, we propose an approach to learn human adaptive control strategy, i.e. how to apply force or torque to accomplish a task according to the task context. By *task context* we mean the dominant factors that will affect the performance of the task, e.g. frictions, object sizes and masses. Because the context is represented directly by the control modules produced by the learning algorithm, the model can generalise to novel contexts within the appro-

priate task domain. The adaptive control strategy is modeled by a modular approach. The key concept of this modular approach is to decompose the adaptive control strategy into several modules often active simultaneously, allowing each to contribute according to its own perceived applicability to the final control command.

From multiple human demonstrations, we extract a set of strategies, each of which takes charge of one specific task context. These strategies include the number of task contexts, which is automatically identified in our approach and also represented by the modules. Each strategy is encoded as a module, which includes a forward model for context estimation, and an inverse model for motor command generation. The forward and inverse models are learnt with a representation that can be easily transferred to a robot. When the robot executes a similar task, the forward models estimate the context of the task and 'contextualize' the inverse models, allowing them to generate the proper commands.

Our work contributes a framework composed of both automated and bespoke components for creating the modular representation of human adaptive control-strategies and to transfer these learnt internal models to a robot. To both communicate and verify our approach, we present two demonstrations: the first a simulation controlling the motion of an object moving through a viscous environment of varying fluid properties, and the second a robot task, *Opening Bottle Caps*. An adaptive control strategy is required for both. For example, for the robot, the friction between the bottle's and the cap's surfaces has multiple phases. In the full robot demonstration, we show how the algorithm modularizes a human-demonstrated control strategy, and how this can be transferred to an arbitrary robot without consideration of the correspondance problem, and demonstrate that the transferred strategy can be used to open both familiar and novel bottles.

The rest of this article is organized as follows: Section 2 provides an overview of related work; Section 3 presents our approach of learning a multiple-module model of a human manipulation strategy. The experiments on the opening-bottle-cap task and their results are shown in Section 5, along with details of the hardware specifications and the experimental setup. Section 6 discusses the proposed method and a look towards future work, followed by the conclusion in Section 7.

2 Related Work

In this section, we give an overview of the area of the machine-learning of manipulation tasks for robots, and of modular approaches.

2.1 Learning Manipulation Tasks

Demonstration-based learning has been extensively studied as a promising approach for building robot intelligence (Calinon et al, 2007; Dillmann, 2004; Kulić et al, 2012). Manipulation tasks are one of the main applications of this approach. The physical properties of a manipulation task are hard to express analytically, and as a result the control strategy is hard to derive. Modeling an expert's demonstration of strategies has been used as an alternative to fully analytical solutions.

Two major forms of demonstration are used in teaching manipulation tasks: kinesthetic teaching and tele-operation. In kinesthetic teaching, a human directly contacts the robot and guides the robot's movements to accomplish a task (Korkinof and Demiris, 2013; Pais and Billard, 2014; Pastor et al, 2011; Li et al, 2014). The trajectory of movements and contact force are recorded by the robot's sensors. This method is simple and effective, but it is limited in the number of controllable end effectors. While a manipulation task usually involves multifinger movement, a human can only operate one finger with each hand and hence two fingers simultaneously at most. To control multi-finger hands, some researchers use tele-operation (Bernardino et al, 2013; Kondo et al, 2008; Fischer et al, 1998). This usually relies on data gloves or other motion-capture systems, which sense the human hand and arm motions. The human motion is mapped to the robot's to generate motions in the robot in real time, allowing the robot to record its own interactions with the environment. In fine manipulation tasks, the robot platforms are usually restricted to anthropomorphic hands for better mapping. Neither kinesthetic teaching nor tele-operation methods provide direct force feedback to the human demonstrator during manipulation. With only visual feedback, it is difficult for the human to conduct manipulation naturally.

Another approach involves the human demonstrating manipulation tasks with their own bodies, rather than directing the robot (Asfour et al, 2008). With direct interaction with the object, the human demonstrator is able to perform the task most naturally and with a more delicate control strategy. However, the task information captured from these human demonstrations must then be transferred to robots. This involves the problem of creating a mapping between the motions of a human and those of a robot, a problem known as the correspondence problem (Nehaniv and Dautenhahn, 2002). Various methods for mapping between human and robot have been proposed (Hueser et al, 2006; Asfour et al, 2008; Do et al, 2011). These may be augmented with correction by humans (Calinon and Billard, 2007; Sauser et al, 2011; Romano et al, 2011) and by self-correction via learning (Huang et al, 2013a). In general, the

effective transfer of human skills to robots remains a challenge.

Our proposed method derives from this last class of demonstrations. We allow the subject to perform a manipulation task directly on an object and experience natural feedback. Our contribution is to encode the strategy in a way that can then be easily transferred to any robot platform. In our task demonstration, a human wears tactile sensors mounted on a dataglove, and directly interacts with objects. The demonstration is recorded and expressed from an object-centric viewpoint. The object-centric viewpoint (Okamura et al, 2000; Jain and Kemp, 2013; Li et al, 2014) centers the representation of the manipulation task on the manipulated object, rather than on the robot. This suggests that the goal of a manipulation task is to produce a desired object movement rather than a robot end-effector movement. Our approach takes this principle and learns a control strategy for producing a desired object behavior. The demonstrated strategy expressed from the object perspective can then be transferred to a robot platform by converting the exerted force to robot joint torque.

With an object-centric viewpoint, we need to learn the correlation between the exerted force of the object and the desired object motion. A classic model of this correlation is impedance (Howard et al, 2010; Wimböck et al, 2012). Given the desired impedance of a task, we can compute proper motor commands for the robot to accomplish it. Fixed impedance control is limited to simple tasks. In many manipulation tasks such as opening a bottle cap, variable impedance is required: at the beginning we need a large impedance to break the contact between the bottle and the cap, and later we need a small impedance to drive the cap smoothly. For such tasks, fixed impedance control will either lead to task failure or cause hardware damage. However, computing the impedance for a given task involving variable impedance is difficult. In many cases the impedance is roughly approximated by a linear model, but this is inadequate for nonlinear tasks.

Variable impedance can be learnt by a human physically correcting the robot's impedance—for example, wiggling the robot's arm—in different stages of the task (Kronander and Billard, 2012). For learning manipulation, however, wiggling the robot's fingers will interrupt the task and may cause task failure. Variable impedance can also be learnt with the reinforcement learning algorithm Policy Improvement with Path Integrals (PI^2) with a task specific cost function (Buchli et al, 2011). Designing this cost function requires insight into the task and usually is difficult. Therefore, in our approach we directly model the correlation of the exerted force and the object motion by a nonlinear statistical model.

In the approaches mentioned above, a single model is built for the entire task. For tasks involving of multiple phases of dynamics, such as are generated by friction, a single control strategy may be inadequate. To handel varying task contexts, robots operating in human-centric environments need to be equipped with multiple strategies. In the next section we give a brief overview of multiple model approaches, that is, of modular approaches.

2.2 Modular Approaches to Learning Manipulation

Modular approaches are widely used in adaptive control and its benefit has been long discussed (Athans et al, 1977; Jacobs et al, 1991; Narendra et al, 1995; Narendra and Balakrishnan, 1997). In manipulation tasks, context changing is a common phenomenon due to object interactions. These changes are often rapid or discontinuous. Classic adaptive control approaches such as model identification (Khalil and Dombre, 2004) are inadequate for these tasks, as instability or error may occur during the optimization of the model variables. To quickly adapt, a modular approach referred to as the Multiple Model Adaptive Control (MMAC, Athans et al, 1977) has been proposed. The paradigm of this method is to design multiple controllers, each of which is in charge of a certain task context. During control, the task context is estimated online and the corresponding controllers are activated. Some authors have relatively recently presented promising modular approaches to solve such control problems (Fekri et al, 2007; Kuipers and Ioannou, 2010). Modular arechitectures have also been shown to be effective for building intelligent systems (Kortenkamp et al, 1998; Bryson, 2000; Bryson and Stein, 2001). In robotics, this approach is particularly useful for tasks in non-stationary environments (Sugimoto et al, 2012).

The modular approach we describe here is inspired by MOSAIC (MOdular Selection And Identification for Control, Haruno et al, 2001). MOSAIC is a paradigm of multiple-module control, where each module is composed of a forward model and an inverse model. The forward models are responsible for estimating the task context in real time, and the inverse models are used to generate appropriate motor commands for the current context. The inverse models are weighted by the accuracy of the estimations of their corresponding forward models. The final motor command is the linear combination of the commands factored by their weights. This paradigm has also been extended to a hierarchical architecture HAMMER (Hierarchical Attentive Multiple Models for Execution and Recognition) to plan high level behaviours (Johnson and Demiris, 2005; Demiris and Khadhouri, 2006). In HAMMER, the lowest level of the hierarchy represents the primitive motions, while the higher-level models represent complex behaviours composed by the primitive motions.

We take the paradigm of MOSAIC but implement the modular model in our own manner. In earlier work, Wolpert

and Kawato (1998) used Artificial Neural Network (ANN) to encode the internal models, i.e. the forward models and the inverse models. The variance of a forward model, which decides how much the multiple modules collaborate, has to be manually tuned. MOSAIC addresses this hand-tuning problem by modeling the transition between modules using a Hidden Markov Model (HMM) and optimizing the variance with the Expectation Maximization (EM) algorithm (Haruno et al, 2001). However, this method requires the forward models to be approximated by linear systems. In order to solve the hand tuning problem of the variance without restricting the complexity of the internal models, we encode our internal models with Gaussian Mixture Models (GMM, Cohn et al, 1996). Training the GMM by the EM algorithm, we compute the optimal values of the models' parameters. GMM has been shown to be efficient for capturing the nonlinearity of data (Calinon and Billard, 2007; Sauser et al, 2011; Huang et al, 2013b). This allows us to approximate more complex internal models. We call each pair of forward-inverse pair a module, hence our system is a multiple module system. Our system focuses on building adaptive control policies for primitive tasks such as opening bottle caps or drawers. Each module both handles and detects an aspect of task context. Multiple modules may be expressed simultaneously, and their outputs integrated. Contexts therefore are not discrete, and some generalisation across and even beyond demonstrated contexts can be performed by the learned model. The combination of the modules thus implements a single task in varying task contexts. This is different from the HAMMER architecture mentioned above, which is for learning high level tasks, combining the primitive tasks hierarchically and forming complex behaviour (Johnson and Demiris, 2005). HAMMER assumes a library of primitive tasks, and an attention mechanism is used to pre-select the high level inverse models, i.e. to only select the inverse models that are required for the task.

In another approach, Petkos et al (2006) have united the forward model and inverse model into a single model. For the particular task described in their paper, i.e. moving a 3-joint arm on a pre-defined trajectory, the action (a_t) taking the current task state (s_t) to the desired task state (s_{t+1}) is always unique. However, in many cases this mapping is not unique and hence the inverse model has to include extra variables in order to resolve the non-uniqueness. To take a more general approach, we build the forward and inverse models separately.

In our model, the final motor command is the sum of the outputs of all modules. This stands in contrast to the switching modular method (Narendra and Balakrishnan, 1997), where only one module will be activated to generate motor commands per time step. Our approach therefore requires fewer modules to approximate the system dynamics.

Despite the many implementations of the modular approach, its application in robotics is not yet wide-spread. One main challenge is how to modularize a given task — that is, how to decompose the task, and how to determine the number of modules. This is a fundamental problem in the research of motion primitives. Kulić et al (2008) use a hierarchical clustering method to extract primitives from human motion sequences. Different cut-off parameters are tested to evaluate the trade-off effects between facilitating quick group formation and introducing misclassification. Our modularized approach is similar to this, but goes one step further. We cluster the demonstration data with a hierarchical method. Instead of hand tuning the cut-off parameter, we determine its value by the variance of the data. This provides us with a proper grouping of the data, which can generate proper motor commands for control.

Figure 3 illustrates the workflow of our approach. To the best of our knowledge, our work is the first realization of the modular approach in learning an object manipulation task with a real robot.

3 Methodology

We have briefly introduced our method in the previous section and justified our design decisions in the light of related literature. In this section we present our method for modularizing human demonstrations of manipulation tasks. Our goal is to acquire a modular control policy for an object manipulation task from human demonstration. To this end, we take a three-step approach:

1. Human demonstration of a task in several different contexts (Section 3.1).
2. Extraction and modular decomposition of human control strategies for different contexts, building multiple internal models (Section 3.2).
3. Robot control using the integrated modules to compute motor commands (Section 3.3).

Figure 1 shows an overview of our framework.

3.1 Human demonstration

The first step is recording the human demonstrations of a task. Based on the object-centric principle, we collect the object's trajectory and the force driving it. We collected this data by a vision-based motion-capture system, force-torque sensor and wearable haptic devices. Figure 2 shows a few of the sensors we used in the opening-bottle-caps task.

In the demonstrations, the demonstrator performs a task a number of times to generate enough data to reliably capture its key features. The demonstrator also performs the task under a variety of conditions, e.g. a range of friction

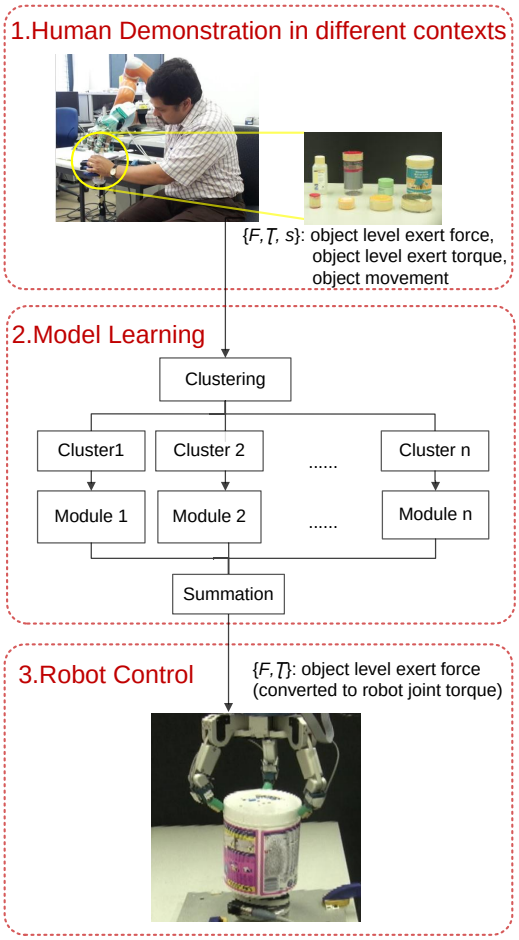


Fig. 1: System overview. Our system takes a three-step approach. 1) A human demonstrates a task in a variety of contexts. In the opening-bottle-cap experiment, the demonstrations are done with different bottles and caps. The object-level exerted forces and torque, and the the object's movements are used for training. 2) Clustering is run over the data from the human control strategies. Each cluster is then modeled as one module. 3) The multiple modules are integrated to compute motor commands to control a robot performing the same task in similar contexts

(a) Optitrack markers attaching to a cap (b) Force torque sensor (c) Texscan tactile sensors mounted on a glove

Fig. 2: Sensors used in the human demonstration of opening a bottle cap task

conditions, in order to explore how humans adapt to different task contexts. These different configurations must be chosen to cover a wide range. For example, in an opening-bottle-cap task, the demonstration of opening the tightest bottle within the capability of the learner is included. This wide range of demonstrations is then used to learn a multiple module model. Details including the exact numbers and durations of our trials are described in the next Section.

3.2 Learning a Multiple Module Model

Here we detail our modeling method, explaining how we model the human manipulation strategy. This requires determining the number of modules to represent a task strategy, learning the internal models for driving each module, and determining how to integrate the output of the modules.

3.2.1 Object centric manipulation strategy

As mentioned in Section 2, one of the challenges in imitation learning is the correspondence problem, i.e. how to map the demonstrator's motions to the robot's motions so that they produce the same effects, such as reaching the same point. In an object manipulation task, the goal is to deliver the object from the current state to a desired state. During this process the movement of the manipulator is bounded by the movement of the object. Thus it is more important to imitate how the human applies force to achieve the object's desired movement than to imitate human limb movement. This is part of what justifies our object-centric representational approach.

The object-centric approach means that our model encodes a force and torque profile rather than the end effector movement trajectory. The imitation-learning objective here is not to find a policy for the end effector movement but to find a policy that maps force and torque to object movements. This policy allows the robot to efficiently acquire new behaviors to accomplish the task. Given the robots' kinematics and the desired exerted force and torque on the object, the robot joint torques can be deduced by their Jacobian matrix (Okamura et al, 2000). To this end, we focus on the force-torque-displacement tuple: $\{F, \tau, s\}$ demonstrated in the task, where F is the exerted force in all directions including the grip force, τ is the exerted torque in all directions and s is the object displacement. In later sections, we refer $\{F, \tau\}$ as the motor command (action) with notation $\{a\}$. In each demonstration, a time series of the tuple is recorded.

We found that cyclic tasks, i.e. tasks that involve a few iterations of similar motions such as rotating an object, further require segmenting the time series. This allows the clustering algorithm to more easily detect repeated or discriminated primitives in the sequence. The problem of automatic segmentation is not in the scope of this work, however possible solutions of it are available in the literature (Demiris and Khadhouri, 2006; Kulic et al, 2009; Pais et al, 2013).

3.2.2 Deciding the number of modules

Due to physical interactions with an object, a manipulation task frequently encounters abrupt changes of the system dynamics, for example transfer between statuses with no contact and with contact, between statuses driven by static friction and by dynamic friction. Different strategies should be

used to handle different dynamics. This motivates our multiple module representation. Our approach is to extract strategies from multiple demonstrations and build one module for each of the strategies.

Different tasks will require different numbers of modules. In the human demonstrations, the same task is demonstrated with a few different setups to explore how humans adapt to them. The number of setups also does not necessarily equal to the number of modules needed in the task. Humans may regard different setups as the same task context and handle them with the same control strategies. In order to find a proper number of modules, we need to differentiate different types of strategies. The differences can be reflected in different patterns of the force-torque-displacement tuple. We differentiate the patterns in a data-driven manner: clustering across the force-torque-displacement tuple. Data in the same cluster is considered to be governed by the same strategy. The number of clusters determines the number of modules.

The goal of clustering is to separate a set of data into a few groups according to their similarities. The first step of clustering is to measure the similarities, i.e. the distances, between different data points. The data we need to cluster are a set of time series: the demonstrations. Here we use the Dynamic Time Warping technique (DTW) to measure the distance between each pair of time series (Berndt and Clifford, 1994). Dynamic time warping is suitable for measuring the similarity between two time series, which may have different speeds or durations. It warps the data in the time dimension and finds the optimal match between the time series. The similarity is computed as the average distance between the corresponding points in two series.

We compute the similarity (distance) between each pair of time series by DTW and get a distance matrix. In this distance matrix, each element contains a measurement of the distance between two time series. We then cluster these time series into a few groups by a threshold of the distance. This threshold is set by using the variance of the data demonstrated under the same condition as a reference. As mentioned above, a task is demonstrated a few times. Demonstrations with the same condition, i.e. the same experimental setup and at the same stage of the task, are presumed to be handled with the same strategy and hence belong to the same cluster. For example, our opening bottle cap task is a cyclic task — humans need to do a few cycles of rotation of the cap in order to unscrew and lift it. In this task, demonstrations with the same bottle and cap, and at the same cycle of the rotation are considered to be in the same group (details in Section 5.2). The variance of these demonstrations give a reference of a proper variance of a cluster. The largest variance, across the variance of all setups, is used as the threshold for the clustering.

Many clustering methods require the specification of the number of clusters. In our case, however, the number of clusters is an unknown variable. Therefore we use the hierarchical agglomerative clustering method (Willett, 1988) to group our data. Agglomerative clustering is a method that merges similar data iteratively until the stop criteria is satisfied — this does not require a predefined number of clusters. Our clustering method only requires the distance threshold of merging and is described as follows:

1. At the beginning, each single time series is considered to be one cluster.
2. Compute the distances between each pair of clusters.
3. Starting from the first cluster, find its nearest cluster. We define the distance between two clusters to be the average distance across all the time series pairs in each cluster. If the distance to the nearest cluster is smaller than the threshold, merge these two clusters. Otherwise leave these two separated.
4. Move to the next cluster. Repeat the last step for the rest of the clusters.
5. A new set of clusters will have been formed by the last few steps. Move to the next level of the hierarchy and repeat the step 2 to 4 until no new clusters can be formed, i.e. no pairs of clusters have distance smaller than the threshold.

Pseudocode of the complete algorithm is shown in Algorithm 1.

Algorithm 1 Agglomerative Hierarchical Clustering

```
Init(): Make each time series a cluster; set the threshold
mergeable = true
function MERGE(all clusters, distance matrix)
    while mergeable is true do
        mergeable = false
        for each cluster do
            ClusterA = current cluster
            ClusterB = nearest neighbor of ClusterA
            if distance(ClusterA, ClusterB) < clustering threshold
then
                Merge ClusterB into ClusterA
                mergeable = true
            end if
        end for
    end while
end function
```

When the clusters cannot be merged further, we define the number of modules for this task: it is the number of the remaining clusters. Each cluster is used as a module. The pattern of the data in a cluster represents a strategy for handling a specific task context.

3.2.3 Learning Internal Models for Each Module

After identifying the number of modules and the data assigned to each, we build models for each module from its associated data. In this section, we explain the way we encode human manipulation strategies using machine learning to build the modules.

During demonstrations, we constantly acquire the object displacements and the force and torque applied by the demonstrator. The demonstrator is the only source of exerted force and torque in the system. The relationship between the exerted force and torque and their resulting object displacement shows the dynamic characteristics of the task.

We model the correlation of the force and the displacement with GMM. The task dynamics is hence encoded as a joint distribution of the object status displacement s and the action a taken by the human, $p(s,a,|\Omega)$. In our experiment, s is the one-dimensional angular displacement of the cap, and a a vector containing the one-dimensional exerted torque and the one-dimensional grip force (Section 5). Modeling their distribution by GMM allows us to capture the nonlinearity in the data, and also to compute the likelihood of a query data point in the model. This provides a good estimation of the reliability of the module in the current task context, which is crucial in choosing the correct modules for control (discussed in Section 3.3.1). Further, as a generative model, a GMM is able to generate new data—that is, it allows us to generate motor commands. This is done by *Gaussian Mixture Regression* (GMR). Table 1 explains the encoding process of GMM and the generative process of GMR.

We aim to build a model that closely emulates the human motor strategy in order to make the best use of the human data. A forward model is held to anticipate the outcome of the motor command, while an inverse model is held to generate motor commands to take the system from the current state to the next state. The discrepancy between the anticipations of the forward model and the actual feedback is used to correct the motor commands generated from the inverse model (Section 3.3.1). Figure 3(a) shows the basic control flow of a forward-inverse model pair, while Figure 3(b) illustrates how three pairs of forward-inverse models work together to generate control commands.

We encode the forward model Ω_F by the joint distributions of the current system state (object displacement), previous system state and the previous motor command, i.e. $p(s_t, s_{t-1}, a_{t-1} \mid \Omega_F)$, and similarly encode the inverse model Ω_I by the joint distributions of the current system state, the desired next system state, previous motor command and the current motor command, i.e. $p(s_t, s^*_{t+1}, a_{t-1}, a_t|\Omega_I)$. The previous motor command a_{t-1} is necessary for the inverse model. In some tasks, the system status can remain unchanged for a certain period until the exerted force reaches a threshold to change it. This will cause degeneracy in the inverse model; hence we include the previous motor command in the model to tackle it.

Table 1: Encoding process of GMM and computation process of GMR

With a Gaussian Mixture Model (GMM), the joint distribution Ω of a set of variables $\{\eta\}$ is expressed as a sum of N Gaussian components:

$$p(\eta \mid \Omega) = \sum_{n=1}^{N} \pi_n p(\eta \mid \mu_n, \Sigma_n) = \sum_{n=1}^{N} \pi_n \frac{1}{\sqrt{(2\pi)^D \mid \Sigma_n \mid}} e^{-\frac{1}{2}(\eta-\mu_n)^\top \Sigma_n^{-1}(\eta-\mu_n)} \quad (1)$$

where π_n is the prior of the n^{th} Gaussian component and the μ_n, Σ_n the corresponding mean and covariance, and D the number of variables.
Gaussian Mixture Regression (GMR) allows us to estimate the conditional expectation value of a variable η^e given a query point η^q where $\{\eta\} = \{\eta^q, \eta^e\}$. To compute this expectation value, first we define:

$$\mu_n = \begin{pmatrix} \mu_n^q \\ \mu_n^e \end{pmatrix} \qquad \Sigma_n = \begin{pmatrix} \Sigma_n^{qq} & \Sigma_n^{qe} \\ \Sigma_n^{eq} & \Sigma_n^{ee} \end{pmatrix} \quad (2)$$

Secondly we compute the expected distribution of η^e from the $n-th$ component:

$$\hat{\mu}_n = \mu_n^e + \Sigma_n^{eq}(\Sigma_n^{qq})^{-1}(\eta^q - \mu_n^e) \quad (3)$$

$$\hat{\Sigma}_n = \Sigma_n^{ee} - \Sigma_n^{eq}(\Sigma_n^{qq})^{-1}\Sigma_n^{qe} \quad (4)$$

Finally, all the N Gaussian components are taken into account, and the expectation value of variable η^e is computed as the mean $\hat{\mu}^e$ with the covariance $\hat{\Sigma}^{ee}$:

$$\hat{\mu}^e = \sum_{n=1}^{N} \beta_n \hat{\mu}_n \qquad \hat{\Sigma}^{ee} = \sum_{n=1}^{N} {\beta_n}^2 \hat{\Sigma}_n \quad (5)$$

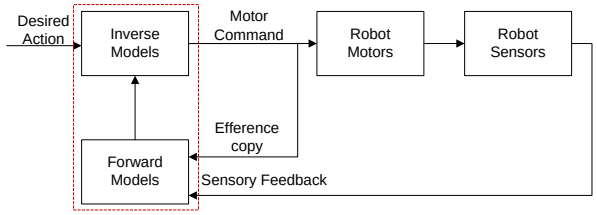
where

$$\beta_n = \frac{\pi_n p(q|\mu_n^q, \Sigma_n^{qq})}{\sum_{n=1}^{N} \pi_n p(q|\mu_n^q, \Sigma_n^{qq})} \quad (6)$$

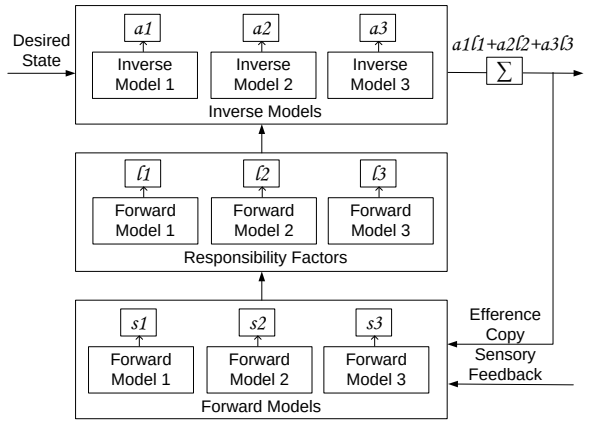
Note that in a multiple module model, different modules may have different numbers of Gaussian components.

3.3 Modular adaptive control and integration

Once the number of modules is found and a pair of forward and inverse models has been learnt for each, the modules can be used to compute motor commands for task execution. In our system of action selection, this process of computing the commands also computes a weight which allows integration of the modules by simple summation. We consider the hu-



(a)



(b)

Fig. 3: Control flow diagram of forward-inverse model in motor control. (a) System overview. Pairs of forward and inverse models work together to generate motor commands. The detailed mechanism inside the red box is shown underneath. (b) An example of a 3-module model. The forward models predict the current task context (s1, s2, s3) and estimate the accuracy of their prediction ($\lambda 1$, $\lambda 2$, $\lambda 3$). These accuracy estimates are called "Responsibility Factors" as they also determine how much responsibility each inverse model should take in the final command. The inverse models generate commands (a1, a2, a3) and the final command is the summation of these, each weighted by its individual responsibility factor (a1λ1+a2λ2+a3λ3).

man motor system acted upon at time t by motor command a_t with current system status s_t. A function f maps a_t and s_t to the system status at time $t+1$:

$$s_{t+1} = f(s_t, a_t) \tag{7}$$

The goal of the controller is to generate a motor command a_t that brings the current system status from s_t to a desired state s^*_{t+1}:

$$a_t = g\left(s^*_{t+1}, s_t\right) \tag{8}$$

In some tasks, the previous action a_{t-1} is included in the inverse model. This is because there may be more than one action (a_t) that can take the current task state (s_t) to the desired task state (s^*_{t+1}) and the previous action a_{t-1} is needed to be taken as reference. Hence the controller needs to be:

$$a_t = g\left(s^*_{t+1}, s_t, a_{t-1}\right) \tag{9}$$

Equation 7 represents the forward model and Equation 8 represents the inverse model. In the modular approach, it takes two steps to compute the motor command a_t:

1. Anticipate the sensory output and compute the responsibility factor λ_t.
2. Compute the motor command of each inverse model and compute the final composite motor command a_t.

3.3.1 Responsibility factor

In a modular approach, choosing the proper modules to control the system at every time increment is a crucial step. For this we rely on a system of *responsibility factors*, which act as the weights of the inverse models. The responsibility factor is a measurement of the reliability of using one module to represent the current system context.

With the k^{th} forward model we can anticipate the current state $\hat{s}^k_t$ by using *GMR* (Table 1):

$$\hat{s}^k_t = E\left(s_t | s_{t-1}, a_{t-1}, \Omega^k_F\right) \tag{10}$$

By comparing the anticipated current state $\hat{s}^k_t$ with the actual current state s_t detected by the sensors, we can evaluate how well the k^{th} module represents the current system. The actual current state, previous state and the previous motor command form a data point $\eta_t = \{s_t, s_{t-1}, a_{t-1}\}$. As the forward models are built as GMM, it is easy to compute the likelihood of one data point belongs to a particular model (the k^{th} forward model): $p(\eta_t|\Omega^k_F)$. The discrepancy between $\hat{s}^k_t$ and s_t is embedded in this likelihood and hence in practice we only compute the $p(\eta_t|\Omega^k_F)$ and skip $\hat{s}^k_t$. The responsibility factor of the k^{th} inverse model is the likelihood of the data point η_t belongs to the k^{th} module, normalized by the total sum:

$$\lambda^k_t = \frac{p(\eta_t|\Omega^k_F)}{\sum_{j=1}^{J} p(\eta_t|\Omega^j_F)} \tag{11}$$

where J is the number of modules.

In the case that the denominator is very close to zero, the whole control process will be terminated as it indicates that the model is used on a different task. At every time step, we compute the responsibility factor for each module. The final motor command at that time step is the linear combination of the commands generated from each inverse model multiplied by its respective responsibility factor.

3.3.2 Generating motor commands by inverse models

The motor command a^k_t for the k^{th} inverse model is computed by GMR with the steps explained in Table 1. At each time step, the responsibility factors λ^k_t weight its corresponding inverse model: the higher the responsibility is, the

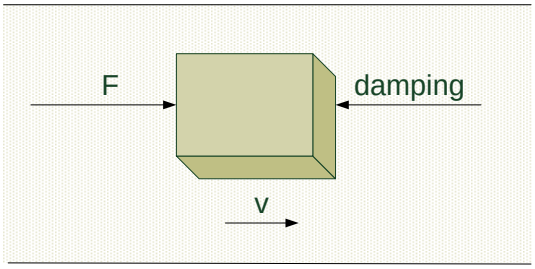


Fig. 4: Illustration of the task. A object moves in an environment with changing damping. This can be understood as moving in a think liquid.

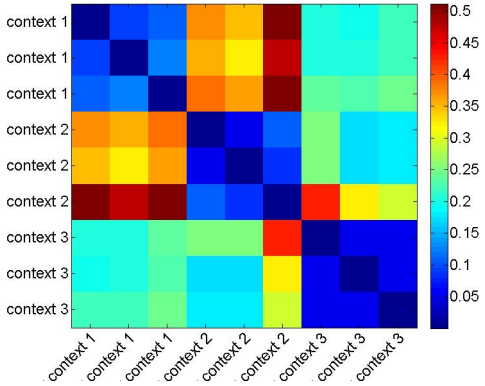


Fig. 5: A heatmap representation of the distance matrix of the 9 sets of training data.

more responsibility the inverse model takes in the control. The final motor command generated by this multiple model system is:

$$a_t = \sum_{k=1}^{K} \lambda_t^k a_t^k = \sum_{k=1}^{K} \lambda_t^k E\left(a_t \mid s_{t+1}^*, s_t, a_{t-1}, \Omega_I^k\right) \tag{12}$$

where K is the number of modules.

These three steps are all computed with a closed form solution. This ensures that this system can react quickly to the changes in the environment by adjusting the responsibility factor.

4 Simulation

In the previous section we described the details of our multiple module approach for learning manipulation strategies. In this section, we evaluate this approach by a nonlinear and non-stationary control task in simulation.

We simulate an object moving in an environment with changing damping, e.g. a thick liquid (Figure 4). The target is to apply force onto the object so that it moves with constant speed. We simulate three different task contexts, i.e. three different damping conditions:

1. context 1: damping = Dv
2. context 2: damping = $D\sin(v)$
3. context 3: damping = $D\tanh(v)$

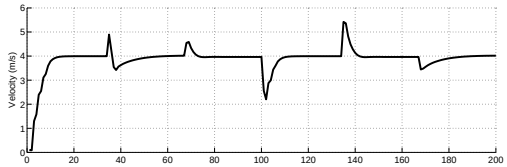
where v is the object velocity and D is the parameter of damping.

During the task execution, the task context randomly switches from one to another. This requires the controller to quickly recognise and adapt to the changes. There is not an easy way to build a single module controller for such an environment without providing an explicit measure of damping as input. We apply our multiple module approach in this task to evaluate its efficiency. The simulation is performed in Matlab. The mass of the object is set to be $5N$, D to be 15 $N.s/m$ and the target velocity is set to be $4m/s$.

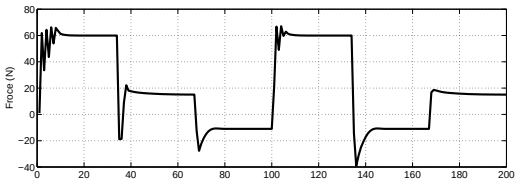
To learn a control policy for this system, we first demonstrate in the different task contexts. For each context, we generate three demonstrations by applying a sinusoidal varying force (F) to the object for a period of time. Randomly generated small noise is added to the force at each time step to simulate natural variability across the demonstrations. During the demonstrations, once the object starts to move, the force, previous velocity, and current velocity $\{F_t, v_t, v_{t+1}\}$ are recorded. After recording a total of 9 sets of demonstrations, we use DTW to compute the similarities between these demonstrations. The distance matrix is shown in Figure 5. As can be seen from the figure, the three task contexts can be clearly distinguished. This shows that by using this approach, different task contexts and their corresponding strategies can be properly separated into different modules.

We hence group the demonstrations into 3 clusters and learn three modules for the task. As mentioned in the previous section, each module is composed of a forward and an inverse model. In this task, the relation between the action and the state is always unique, hence we encode the forward and inverse models by the same GMM $\{F_t, v_t, v_{t+1} \mid \Omega\}$. With the forward model, we computed the expectation value of the next velocity $E\{v_{t+1} \mid \Omega, F_t, v_t\}$ by using GMR. The responsibility factors of each forward model are then computed according to the Equation 11. Finally, the motor commands are computed by the linear combination of the inverse models according to the Equation 12.

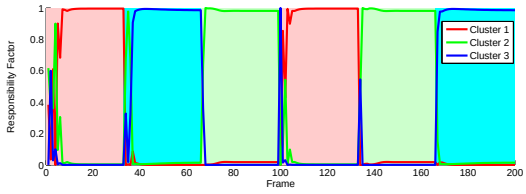
We apply the above mechanism to move the object. The damping of the environment is constantly switching across conditions. Figure 6 shows the results. As can be seen from the figure, when the task context switches, the forward models can quickly recognize the correct context and hence guide the inverse models to produce the proper command to maintain the object's velocity. After the switches, the object's velocity can quickly be corrected to the target value. This simulation experiment shows that the proposed ap-



(a) Object velocity. The target velocity is 4m/s



(b) Force applied to object



(c) Responsibility factor for each module during task execution. The background colors represent the actual task context: pink-context 1, light blue-context 2 and light green-context 3.

Fig. 6: Simulation results of moving an object in a changing environment

proach can indeed properly recognize the current task context and generate fast adaptive motor commands.

The full Matlab code for this demonstration is provided as an electronic appendix.

5 Robot Experiment

In the previous sections we have described the details of our multiple module approach to manipulation task learning in a generic way, and demonstrated it in a relatively simplifed though non-linear and non-stationary simulation task. In this section, we describe the experimental details for our application to the bottle-opening task. We demonstrate that the multiple module approach is able to acquire a human adaptive control policy and enable the robot to master this manipulation task.

We implemented our multiple module approach on a real robot system consisting of a 7 DOF Light Weight KUKA robot arm[1] with a 4 DOF Barrett Hand[2] The target manipulation task is to unscrew a tightened cap until it can be lifted from its bottle. This task is chosen because it is a common task in human daily life, and at the same time a complex task from the control point of view. The friction between the bottle and the cap plays an important role in the task: it largely determines the exerted torque required to open the cap. However, the friction, and the way it changes as the cap unscrews, varies between different bottles.

Estimating the friction coefficient (FCO) solely according to the material is difficult, as it is affected by many factors such as the load force, movement velocity, contact surface situation, composition of the material, temperature, etc. (Gustafsson, 2013). A deterministic control strategy based on the value of the FCO is not practical in this task. A small estimation error in the FCO may produce either too small a torque, which leads to task failure, or too large a torque, which may cause hardware damage. Therefore an adaptive control strategy is desired for this task. We use our multiple module approach to model the adaptive strategy.

5.1 Human demonstration and experimental setup

Opening a bottle cap is a common task for human but not an easy one for robot. Before the task begins, the human does not possess any information about the tightness of the cap. This information can only be estimated once the task is started. During the task, a human will constantly update the motor commands, i.e. how much torque to apply to the cap and with how much force to grip the cap, according to the sensory feedback. This plan can only be made in real time as the contact surface condition changes throughout the task process. Humans have to cope with these uncertainties and adapt to the changes. Figure 7 shows three different patterns of human control strategies for three different contexts. This task requires an adaptive strategy that controls the turning torque, gripping force and the displacement of the cap. Learning from human demonstration allows us to gain such a control strategy without fully analyzing the dynamics of the whole system.

In each demonstration, data is recorded from the first time a finger touches the cap to when the cap is finally open and lifted. Opening a bottle cap is a cyclic task. Each cycle includes three stages: reaching, turning and releasing. In our experiments, four to six cycles need to be completed to open the bottles. During the reaching and releasing stages, neither torque nor gripping force is applied to the cap and the cap remains still. During the turning stages, humans continuously apply torque to the cap and it starts moving once the friction is overcome.

5.1.1 Demonstration in different task contexts

In order to explore different task contexts, we demonstrated the task with different setups, which are the combination of four different plastic bottles ($b1-b4$) and four different plastic caps ($c1-c4$) (Figure 8). The caps are made

[1] http://www.kuka-labs.com/en/medical_robotics/lightweight_robotics
[2] http://www.barrett.com/robot/products-hand.htm

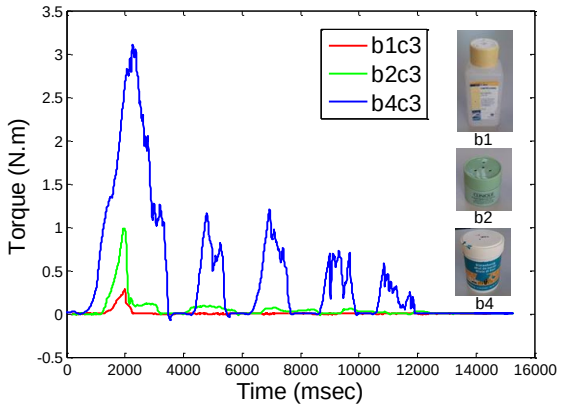


Fig. 7: Exerted torque for opening three different bottles. These three demonstrations show different control patterns. Significant differences can be observed from the first cycles. For the most difficult bottle b4 (blue), the torque reaches its peak and drops slowly until the end of the cycle. This shows the friction between the bottle and cap keeps decreasing as the cap starts moving. For the easier bottle b2 (green), the torque drops straight from its peak to a (relatively) constant value, which suggests that the friction remains constant when the cap is moving. For b1 (red), the torque slowly climbs to a peak, then drops sharply for a short while and then more slowly. These different patterns indicate the different types of friction governing different contexts.

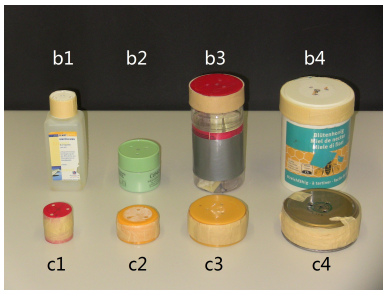


Fig. 8: Bottles and caps for human demonstrations. From left to right: b1 c1, b2 c2, b3 c3, b4 c4

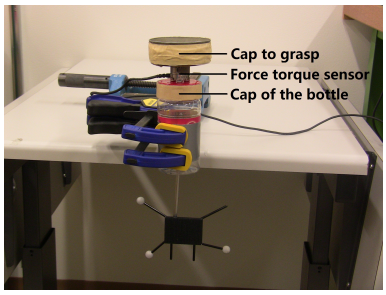
Table 2: Different setups of bottles and caps for demonstration. Bottles 1 to 4 are in increasing order of the difficulty to open. Caps 1 to 4 are in increasing order of the cap sizes, with diameters are shown. Note that where multiple caps were used with the same bottle, this was achieved by affixing the cap to the bottle's matching cap (see Figure 9).

	 Cap 1 25*mm*	 Cap 2 42*mm*	 Cap 3 56*mm*	 Cap 4 80*mm*
 Bottle 1			b1c3	
 Bottle 2			b2c3	
 Bottle 3	b3c1	b3c2	b3c3	b3c4
 Bottle 4			b4c3	

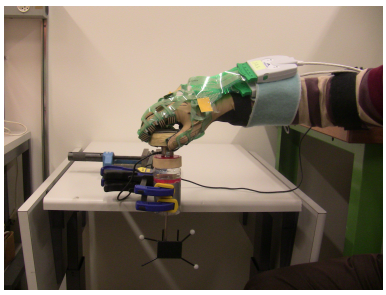
of light materials such that their masses and inertias can be neglected in the computation of motor commands. The relationship of driving force/torque and mass/inertias is well established. In this study, we focus on learning the strategies for handling friction, which is not well understood and hard to predict. According to the surface conditions of the bottles and the caps, the difficulty of opening the bottles varies. $b1-b4$ are labeled by increasing difficulty. The bottle $b1$ is the easiest one, which originally contained body lotion. We lubricated bottle $b1$ with its body lotion to make it even easier. The bottle $b4$ is the most difficult one; it originally contains honey which is very sticky. We left honey on the surfaces of $b4$ to make it more difficult. The difficulty is estimated qualitatively. It is judged according to the friction coefficient between the contact surfaces. Generally speaking, the friction coefficient between lubricated surfaces is smaller than between dry surfaces, while between smooth surfaces is smaller than between sticky surfaces[3]. The $c1-c4$ are labeled by the increasing diameters of the caps.

We chose to vary the setups in surface condition and cap size as these are the main points of variation between the different bottles affecting the control strategy. The intention is to see how these two variables affect human behaviour. To this end, we combine the bottles and the caps by mounting the caps $c1-c4$ onto the 'actual' (manufactured) caps of the bottles (Figure 9). To investigate the effects of different caps and different bottles separately, we conducted two groups of demonstrations: a fixed bottle with four different caps $(b3c1, b3c2, b3c3, b3c4)$ and a fixed cap with four different bottles $(b1c3, b2c3, b3c3, b4c3)$. Demonstrations on the first group allow us to explore human grasping strategies with different cap sizes. Demonstrations on the second group allow us to explore human control strategies in adapting to different bottle conditions. In total, we have seven different setups for the human demonstration (Table 2).

[3] The precise value of the friction coefficient between plastics varies by type of the plastic. According to an Internet resource (Tribology-abc.com, 2014), the dry dynamic friction coefficient between plastic-plastic surface is 0.2-0.4 and the lubricated dynamic friction coefficient is 0.04-0.1.



(a)



(b)

Fig. 9: Experimental setup for the task of opening a bottle cap. (a) Setup b3c4: bottle 3 combined with cap 4. A force-torque sensor is mounted between the 'cap of the bottle' and the 'cap to grasp' ($c4$), so that the exerted force and torque can be measured. A set of Optitrack markers are connected with the cap to record its displacement. The bottle is fixed on a table. (b) A human demonstrating how to open a bottle cap. To avoid extra torque, only one hand is used during the demonstration. The human grips the cap from the top and applies torque to the system

5.1.2 Sensors

In each setup the demonstrator demonstrates the task of opening the bottle cap three times. Before each demonstration, the bottle is tighten with the cap with the same scale of tightness. In total we recorded 21 sets of demonstrations. In this section, we describe the sensor recording of these demonstrations.

As explained in section 3.2.1, we focus on the tuple $\{\tau, F, s\}$ of the task. Three different sets of sensors are used in the experiment to capture them:

1. Force torque sensor[4] for exerted torque (τ);
2. OptiTrack[5] for cap displacement (s);
3. Tekscan[6] for exerted force (F).

Data from these three sensors stream from three different channels. Due to hardware limitations, the raw data stream from the different channels does not come at the same time, and cannot be recorded at a regular frequency. To synchronize the data, we produce a synchronization signal at the beginning of each demonstration: the demonstrator taps on the cap three times. The movement of the hand and impulses on the cap produce simultaneous pulses in all three channels. After recording, the data from the different channels is synchronized by aligning the synchronization signal.

In this task, the turning torque is the essential variable. This is measured and recorded by an ATI force torque sensor. It is mounted between the bottle and the cap (Figure 9). During the task, the demonstrator grasps the cap on the top of the force-torque sensor and applies torque to open the bottle mounted below the sensor. As the bottle is fixed to the table, the movement of the cap is restricted to the rotation along the bottle's axis. Under the approximation of zero angular momentum, the reading of the sensor shows the force and torque applied to the cap. Besides the torque, force applied to the z-axis direction is also recorded for the purpose of synchronization (Section 5.2).

We track the displacement of the cap by a motion tracking system OptiTrack. The OptiTrack system tracks movement by the infrared reflecting markers attached to the object. In order to avoid obstacles tothe demonstration, we attach markers to a stick, which is fixed to the cap from one end and the other end coming out from the bottom of the bottle (Figure 9). We also recorded the human hand movements, by tracking markers attached to the human's hand. The movement of the human hand is used later for synchronization (Section 5.2).

During the task, the human also applies grip force on the cap in order to grasp it firmly for turning. This force cannot be sensed by the force torque sensor. Therefore, we used a pressure sensor (Tekscan Grip System) for measuring the grip force. The Tekscan Grip System is a flexible tactile pressure sensor that can be built into a glove. It has 18 patches of sensors to cover the human's hand's front surface. For manipulation, humans use not only the front surface, but also the side surface of our fingers. In order to measure the force applied by those surfaces, we mount two sets of Tekscan Grip System sensors onto a glove to cover also the side surfaces (Section. 5.2). The method of mounting the sensors to the glove is detailed by de Souza et al (2014).

With different sizes of caps or in different stages of the task, the way a human grasps the cap may vary. For example, a human may use two fingers to grip the smallest cap $c1$, and four fingers to grip the biggest cap $c4$. The patches receiving contact in each grasp are recorded. In the computation of the total grip force, only the patches used are taken into account. All patches are calibrated to give readings in the unit of $N{\cdot}m$.

5.2 Data Analysis

In this section we explain how we manage the raw data and extra training data. The raw data from the three sensors streams is in three separate channels. Each stream has a different format and hence is handled differently.

[4] https://www.ati-ia.com/
[5] http://www.naturalpoint.com/optitrack/
[6] http://www.tekscan.com/

– **Exerted torque** As the movement of the cap is restricted to rotation around the z-axis, we are concerned only with the torque applied in this direction. Another dimension of concern is the force applied in the z direction. The three taps on the cap before each demonstration create three pulses in the z direction and hence is used for synchronization.
– **Object displacement** From the OptiTrack, the cap's displacement is originally expressed in the position vector and the rotation matrix. The angular displacement of the cap is computed by the rotation matrix of the cap, and the hand movement by the position vector of the hand. The accumulated angular displacement is used to learn the model and the hand movement is used to synchronize the data.
– **Grip Force** As mentioned in previous section, we used two sets of Tekscan to cover the front and the side of the human hand. This enables the demonstrator to use any grasp they like for the task — the human was not restricted to using just two or three fingers as is the case in most other grasping experiments. For each type of grasp, the reading from the patches contacting with the cap are summed and multiplied by their surface area to compute the total grip force.

Data from these three channels is synchronized by aligning the synchronization pulses. The time of the last detected pulse is set as the zero-reference point. After synchronization we re-sample all the temporal sequences to 1000Hz. Thus each single data point is synchronized. Finally, we filter the noise by a low pass filter (200Hz). Figure 10 shows an example of the data from three different channels.

In this task we focus on the turning stage of each cycle. More specifically, we focus on the data starting from the moment that the fingers contact the cap and ending at the moment that the turning is finished and the cap is released. The reaching and releasing cycles do not involve contact with the environment and hence are not addressed here.

In order to collect data from only the turning cycles, we trim the data by the contact signal: only parts of the sequence with non-zero contact force are kept[7]. The trimmed sequences are labeled by their associated equipment setup and the order in which they occur, e.g. the first cycle of the bottle 1 with cap 3 is labeled by $b1c3\_1$.

As can be seem from Figure 10, there are dramatic difference between cycle one (the first cycle) and the rest of the cycles: the exerted force and torque are much higher in the first cycle. This is caused by the difference between the static friction and the kinetic friction of the bottles. At the beginning of the task we have to first break the contact between the bottle and the cap. The friction we need to break

[7] In this task the segmentation is done manually. The data can also be segmented by other algorithms but here we do not focus on task segmentation; cf. Section 3.2.1.

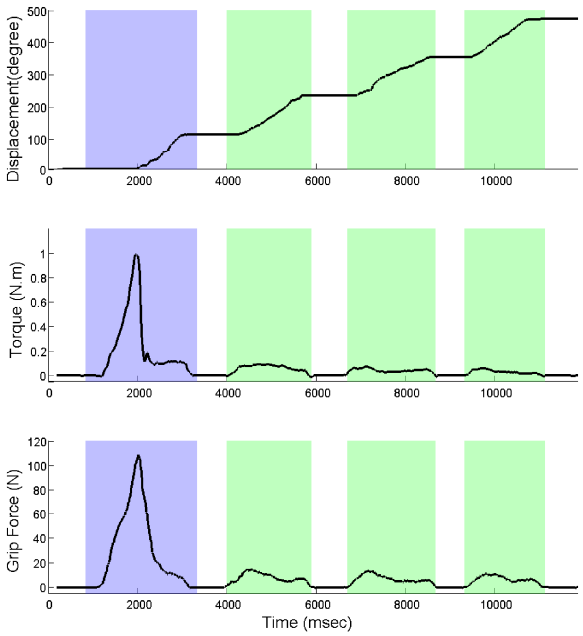


Fig. 10: Aligned data of all three channels. Highlighted parts mark the turning process: blue blocks denote the first cycle (phase I), and green blocks denote the later cycles, (phase II). Phase I is significantly different from phase II

at this stage is determined by the static FCO. Once the cap starts to move, the FCO between bottle and cap transitions to kinetic FCO, which is usually smaller than the static FCO for the same surface conditions. As a result, the torque and hence the grip force required to turn the cap decreases in the later cycles. This phenomenon implies that at lease two modules are needed for this task. In the later section we will discuss these two phases separately and referring the cycle one as "phase I" and the later cycles as "phase II".

In different demonstrations, the number of cycles used to open the cap is different, varying from four to six. The pattern of the later cycles are similar because the demonstrator just repeats the same strategy for rotating the cap. For training, we take the first four cycles from each of the demonstrations. As mentioned above, ahuman demonstrates the task in seven different setups, each for three times. This results in 84 time series in total for the learning.

5.3 Learning Modules

In this section, we explain how we encoded the training data into just a few different modules. As explained in Section 3.2, the first step is to cluster the data and find out the number of modules required in this task.

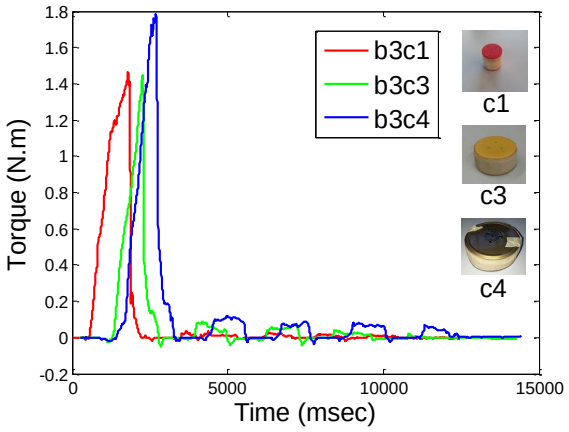


Fig. 12: Exerted torque for opening bottle $b3$ with three different cap sizes

5.3.1 Data clustering

To cluster the 84 time series $Q\{s,\tau,F\}$ obtained from human demonstration, we first compute the distance between each pair of series by the DTW technique. As this task is time independent, "warping" of the data in the dimension of time does not affect the control policy encoded in the time series. The distances between each pair of the time series is shown in the heatmap (Figure 11). As can be observed from the heatmap, the trials with the same setup and in the same cycle are very similar to each other. Hence we regard these trials as representing the same control strategy and use their variance as the threshold of the clustering. Trails with distance less than this threshold are considered to be in the same cluster. This is to say, our clustering is based on the assumption that trials with the same setup and in the same cycle are governed by the same control strategy. This assumption helps us cluster the large amount of data to a small number of modules.

From this heatmap we can also see that within the same cycle, the trials with the same bottle but with different caps, e.g. $b3c1, b3c3$ and $b3c4$, are similar to each other. In the first cycle, the trials with the same cap but with different bottles, e.g. $b1c3, b2c3, b3c3$ and $b4c3$, are significantly different from each other. In the later cycles, this difference decreases gradually. This result shows that in the opening-bottle-cap task, the surface condition between the bottle and the cap plays an important role in the control strategy, while the role of cap size is relatively minor. Figure 12 shows three trials of opening bottle $b2$ with different sizes of caps. It can be seen that their patterns are similar.

As mentioned before, the demonstration of each setup is repeated three times. Those three time series from the same setup and same cycle are presumed to belong to the same group. To set a threshold for clustering, we check the distances between the three time series in the same group. The largest distance we found is 0.04 (normalized) from the $b3c2$ phase 4. We add a 10% margin on this (resulting to 0.044) and use it as the threshold of clustering. Time-series distances less than the threshold are grouped into the same cluster. We use hierarchical agglomerative clustering (Section 3.2.2) to merge the data into fewer clusters. After five mergings, the clusters are no longer mergeable and three clusters remain.

These three clusters contain the data from:

1. phase I of $b4c3$ (most difficult bottle), 3 time series;
2. phase I of $b3c1, b3c2, b3c3, b3c4, b2c3$ and phase II of $b4c3$, 24 time series;
3. phase I of $b1c3$ (easiest bottle) and phase II of the other setups, 57 time series.

The result of clustering is shown in Table 3. This result suggests that humans use three different strategies for opening bottles: one for handling phase I of the most difficult bottle with adhesive materials on the bottle and cap surfaces; one for handling phase I of most bottles and phase II of the most difficult bottle; and one for handling phase I of the lubricated bottle and phase II of the other bottles. The size of the cap turns out to be play a less important role in the control strategies. According to these results, we encode these three clusters separately.

5.3.2 Learning Modules

We encode the data in each of the modules by means of GMM. As explained in Section 3.2.3, a forward model and an inverse model are built for each module. The forward model is encoded by the joint distribution $p\{s_t, s_{t-1}, a_{t-1} \mid \Omega_F\}$, while the inverse model is encoded by $p\{s_t, st+1, a_t, a_{t-1} \mid \Omega_I\}$. For each model, the number of Gaussians is determined by the Bayesian information criterion (BIC). We use 25 Gaussian for cluster 1, 40 for cluster 2 and 15 for cluster 3. The BIC tests for each module / cluster are shown in Figure 13.

5.4 Generating robot motor commands for manipulation

Our approach is independent of the robot system and can potentially be applied to any robot. We chose to implement this work with a Barrett hand mounted on a KUKA lightweight robot arm as these were available in our lab. We implemented the multiple module system on this platform to enable the robot to open bottle caps.

In this experiment, we control the wrist joint (last joint of the KUKA) for producing torque to turn the bottle cap. A force torque sensor is fixed under the bottle to provide torque feedback. Each finger of the Barrett hand is mounted with a

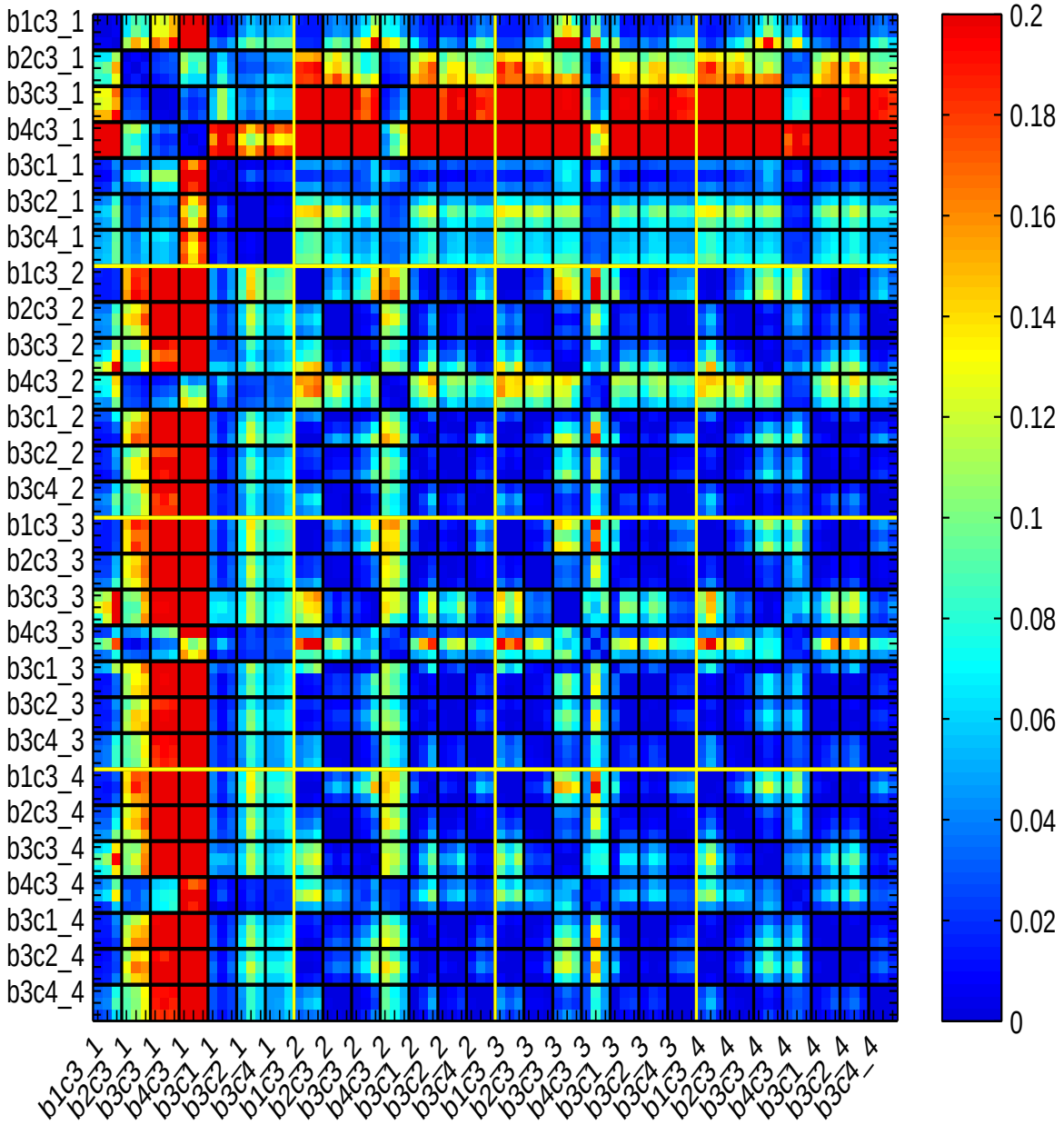


Fig. 11: A heatmap representation of the distance matrix of 84 time series (7 setups $\times$ 4 ycles $\times$ 3 trials). The labels are in the format of "setup_cycle". For example, "b1c3_1" represents the first cycle of the *b*1*c*3 setup. The yellow lines divide the x and y axis by the 4 cycles and hence form 16 large blocks. In each block, the black lines divide the x and y axis by the 7 experimental setups and hence form 49 smaller blocks

Syntouch[8] tactile sensor, which is calibrated to provide contact force information, for the grip force feedback. The cap displacement is measured by the wrist joint displacement, assuming that there is no slippage between the fingers and the cap.

The target bottle is fixed to the top of a table with its cap tightened. The robot is placed above it at a distance that allows a proper grasp on the cap. The Barrett hand then closes the fingers until the bottle cap is touched. This position is recorded as the initial position, where the cap displacement is marked as zero. In the experiment we focus on the turning cycle. The releasing and reaching cycles are programmed by

[8] http://www.syntouchllc.com/

Table 3: Clustering results

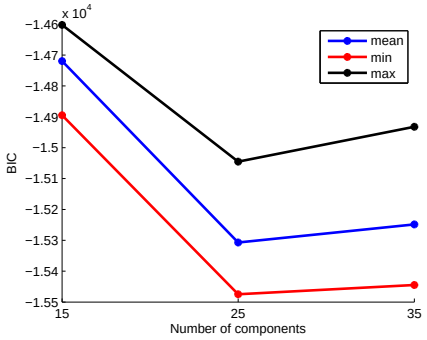
		 Cap 1	 Cap 2	 Cap 3	 Cap 4
 Bottle 1	Phase I			(b1c3) Cluster 3	
	Phase II			Cluster 3	
 Bottle 2	Phase I			(b2c3) Cluster 2	
	Phase II			Cluster 3	
 Bottle 3	Phase I	(b3c1) Cluster 2	(b3c2) Cluster 2	(b3c3) Cluster 2	(b3c4) Cluster 2
	Phase II	Cluster 3	Cluster 3	Cluster 3	Cluster 3
 Bootle 4	Phase I			(b4c3) Cluster 1	
	Phase II			Cluster 2	

Algorithm 2 Control Algorithm

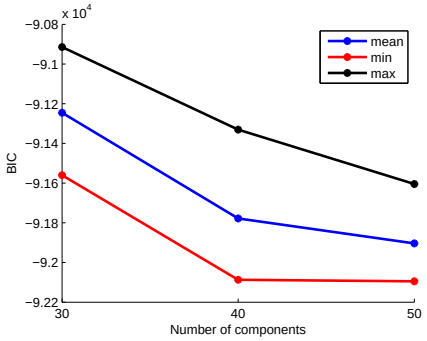
for r = 1:4 **do**
  REACHING(): Robot moves to the initial position
  function TURNING()
    Read previous sensor information $\{s_{t-1}, \tau_{t-1}, F_{t-1}\}$
    for k=1:3 **do**
      $\hat{s}^k$ = FORWARD$(s_{t-1}, T_{t-1}, \Omega_I^k)$
    end for
    for k=1:3 **do**
      λk = ResponsibilityFactor$(\hat{s}^k, s_t)$
    end for
    Read current sensor information $\{s_t\}$
    for k=1:3 **do**
      $\{a^k\}$ = INVERSE$(s*_{t+1}, s_t, a_{t-1})$
    end for
    $\{a_t\} = \sum_{k=1,2,3} \lambda k \{a^k\}$
    Add compensating torque to τ_t
    Execute motor command $\{a_t\}$
    RELEASING(): Release the cap;
  end function
end for
while LIFTCAP() is false **do**
  REACHING();
  TURNING();
  RELEASING();
end while

opening the fingers and restoring to the initial position. The information gathered from the Tekscan during the demonstrations is transferred to the Syntouch and used to guide the grip force. During the process, the tactile sensors are used to ensure the fingers are touching the cap and providing enough grip force to avoid slippage between the hand and the cap.

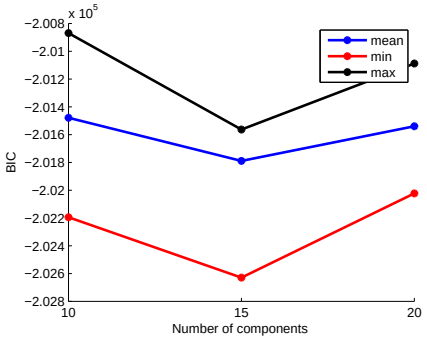
We first tested the model with the trained bottles and then with two new bottles. With each bottle, the turning–releasing–restoring cycles are repeated four times. Data streams from the sensors are filtered to $100Hz$. Once the turning cycle starts, the forward models take the torque and displacement at the last time step as input, compute the expected displacement of the current time step. These expected displacements are compared with the actual displacement measured at the sensor to evaluate the reliability, expressed as a normalized responsibility factor (λ_t^k), of each module (k). The inverse models take the current displacement, desired next displacement, and the previous torque as input to compute the a proper action (torque) to take on the cap. Each of the three outputs is weighted by multiplication with its responsibility factor, and the final output is the sum of the three weighted outputs (Algorithm 2).



(a) Cluster 1. Optimal number of Gaussians is 25.



(b) Cluster 2. Optimal number of Gaussians is 40.



(c) Cluster 3. Optimal number of Gaussians is 15.

Fig. 13: BIC test results for clusters, determining the number of gaussians used in each module

In the process of implementation on a real robot, we found that without putting any restriction on the responsibility factor, it can change very rapidly. This is caused by the environmental noise in the sensory input and results in instability of the control system. We therefore apply a low-pass filter (100 Hz) on the responsibility factor λ_i^k to reduce the fluctuations. This filtering implies that the real dynamics do not switch back and forth with high frequency, which is consistent with the character of our task.

Before applying the final output on the robot, a compensational torque is added to it in order to compensate the lag causing by the distortion of the robot hand during turning. The control algorithm described above is shown in Algorithm 2.

5.5 Experiment results

We validated the algorithm to control cap opening by our robot. We first tested the ability of the system to open two of the bottles seen during training ($b1$ and $b4$). We then tested the generalization capacity of the system by opening two bottles ($b5$ and $b6$) not seen during training. Bottle $b1$ and $b4$ are the easiest and most difficult bottle to open in the training set. Bottle $b5$ is a large bottle, which is hard for human to grasp and open. Bottle $b6$ is a glass bottle with a plastic cap. The surface interaction between these two materials had not been demonstrated. As the Barrett hand is significantly larger than a human hand, $b1, b4, b6$ are mounted with $c5$ (the cap of $b5$ with diameter $110mm$) on the top to ensure a firm grasp. In total, four different setups were used in the experiment: $b1c5, b4c5, b5c5$ and $b6c5$. As discuss above, the size of the cap has minor effect on the control strategy. Therefore we expected the setups $b1c5$ and $b4c5$ to result in similar behavior as those of $b1c3$ and $b4c3$ in the training. The experimental results and demonstration snapshots are shown in figures 14–17[9]. Figure 18 is a similar plot to the figure from the demonstration, figure 7, which aligns the exerted torque of the four experiments.

In each experiment we record the cap displacement, exerted torque, and the responsibility factors of all three modules. Bottle $b1$ is the easiest bottle to open in the training set, the control policies of both phase I and phase II are grouped into cluster 3. As a result, in the $b1$ experiment, module 3 takes most of the responsibility (Figure 14).

Bottle $b4$ is the most difficult bottle to open in the training set and its phase I requires more than $3Nm$ (Figure 7). Due to the smooth contact surfaces between the Barrett hand and the cap, it is difficult to apply $3Nm$ torque to the cap without slipping. To avoid damaging the robot, we tested $b4$ phase II only: the cap is loosely screwed on the bottle. Without knowing this, in the experiment the robot is able to properly estimate the current task context. As can be seen from the figure 15, which is different from $b1$, the dominant module is module 2 which corresponds to $b4$, phase II. This performance would be hard to achieved with a deterministic system based on expected values for friction coefficients.

Bottle $b5$ is a novel one but is made of a similar material (plastic) to the training bottles. A very similar torque profile to $b2$ and $b3$ is generated for $b5$: phase I is sharp, while phase II is flatter and significantly smaller than phase I ($b2$: Figure 7, $b3$: Figure 12, $b5$: Figure 16). This is because $b5$

[9] Demonstration videos are available at http://www.cs.bath.ac.uk/ bh325/opencap.rar and will be provided as an electronic supplement to the article.

has a dry contact surface as $b2$ and $b3$, and $b1$ is lubricated and $b4$ is attached with a sticky material, honey.

Bottle $b6$ is also a novel one but with novel surface materials (plastic and glass[10]). Its torque profile is different from what we observed in training set. Despite this, $b6$ is opened with this torque profile generated by the three learnt modules.

5.6 Discussion of results

In these experiments, the combination of modules applied was determined completely algorithmically, with no human intervention. This is made possible firstly by the variance of the forward models. The variance of a GMM represents the variance of the training data, i.e. human demonstrations, and a GMM can have locally large variance. In our approach, each GMM is learned independently, without knowing the boundaries of other modules. This can result in a GMM having a non-zero probability at the borders of other modules. Secondly and even more importantly, the computation of the responsibility factor as a real rather than a binary value encourages cooperation between modules. The mixed activation of multiple modules allow the algorithm to generalise better to novel task contexts, as more modules will contribute to generate new motor commands. As can be seen from the results, our method can generate commands properly to accomplish the task in both trained and novel task contexts. Other ways of computing the responsibility factor that bring less coordination may cause instability when generalised to new task contexts.

With the above four different setups, the modular model adapts accordingly and successfully generates torque commands to open the bottles. Successful cap opening is achieved when the cap is unscrewed far enough that it can be lifted up. Though no prior information is provided about the bottles, the task contexts are properly estimated and "contextualized" motor commands are generated to unscrew the caps. These experiments show that our multiple modular approach is indeed effective in manipulation tasks.

6 Discussion

In this article we have presented a modular approach for learning manipulation tasks from human demonstration. We discover the number of modules needed in a task by hierarchical clustering. From each cluster we use forward and inverse model pairs to model the motor control mechanism.

[10] A common way of measuring the FCO of a material is measuring it against metal: the static FCO between glass and metal is 0.5–0.7, while between two polythene and steel is around 0.2. This implies that the plastic and glass are indeed very different in FCO. There is not a universal measurement of the FCO between plastic and glass.

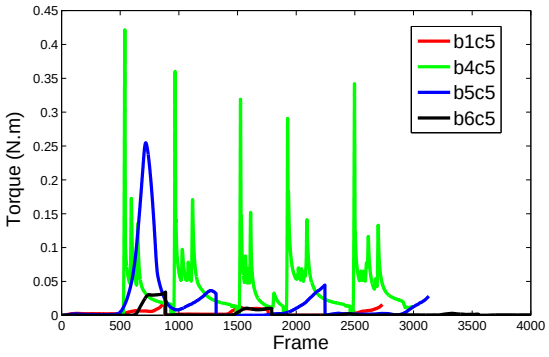
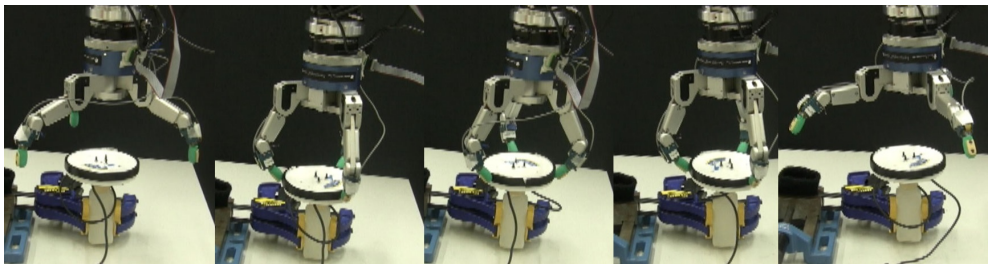


Fig. 18: Robot exerted torque for opening four bottles: b1, b4, b5, and b6. Time is warped and shifted so that the cycles are aligned for visual comparison.

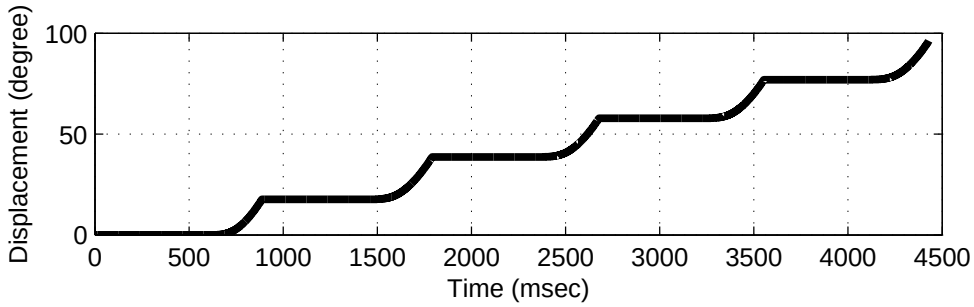
The forward models predict the effect of the previous motor command, while the inverse models compute a motor command to bring the current state to a desired state. The statistical approach enables us to estimate the reliability of the inferences of each module under the current task context. The final motor command is the sum of the weighted commands generated by each module. By exploiting an object-centric viewpoint, the learnt human internal models can be easily transferred to a robot. Our experiments verify that by this modular approach, the robot can automatically recognize the current task context and compute proper motor commands to accomplish a manipulation task, here moving a simulated object through shifting fluid dynamics, and opening bottle caps with a real robot, including on novel bottles.

Our approach is applicable to manipulation tasks that require adaptive control strategies. It has a number of benefits compared to existing, pervasive methods for adaptive control such as classic model identification adaptive control and reinforcement learning (Narendra et al, 1995; Khalil and Dombre, 2004; Buchli et al, 2011). Because we imitate human behaviors, we do not need to derive the system dynamics nor the cost function of the tasks, which involve deep insight into the task and can be painstaking. The difficulty of modeling an adaptive strategy is further reduced by a modular approach: dividing the large state space into several subspaces, where the local strategies can be approximated more accurately. With this approach, we divide a complex human strategy into a few modules, and combine them to generate contextualized motor commands.

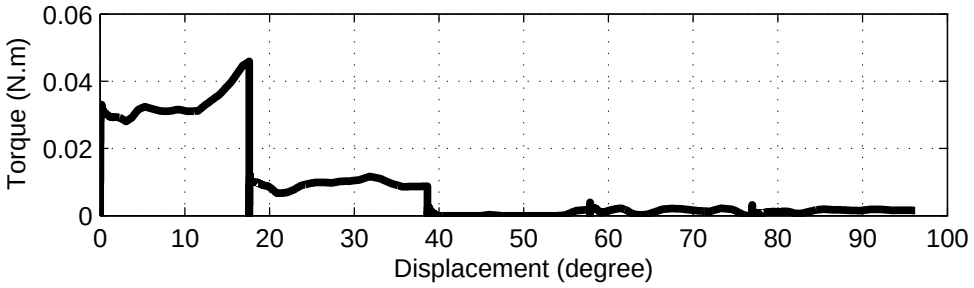
Our object-centric approach is a practical approach for teaching a robot manipulation tasks that require proprioception. This allows human demonstration of the task with physical contact with the object, which means the demonstrator can have direct feedback from their own senses and perform the task naturally. We bypass the problem of direct mapping of human movement and degrees of freedom to a robot's by expressing the strategy from an object-centric viewpoint. This can greatly benefit learning manipulation



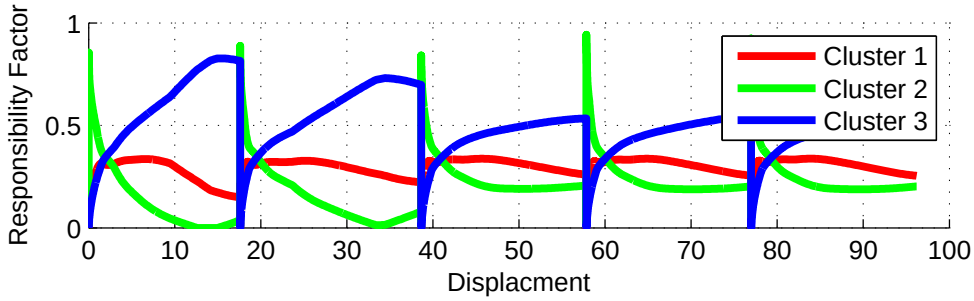
(a) Snapshots from the robot opening bottle $b1$



(b) Cap displacement during the robot's opening

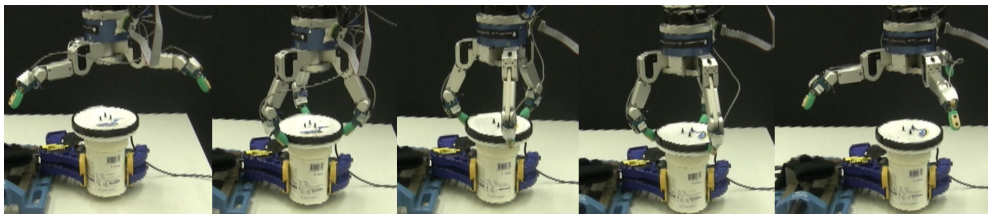


(c) Torque exerted by the robot against cap displacement

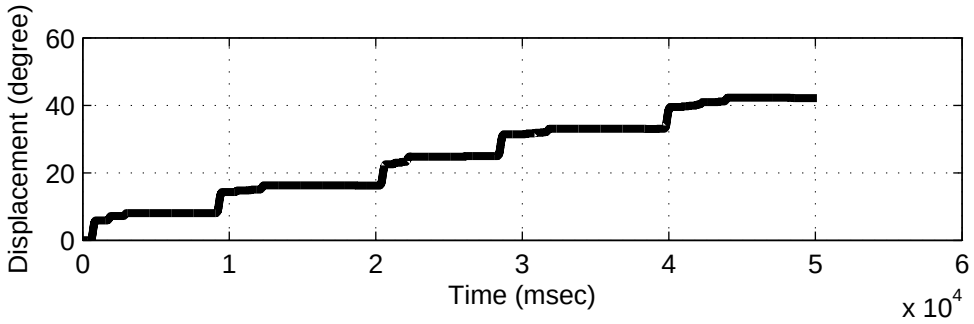


(d) Responsibility factor against cap displacement, for each module

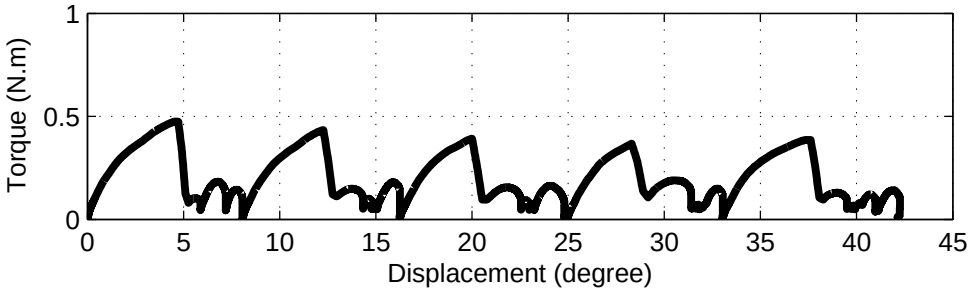
Fig. 14: The robot opens bottle $b1$.



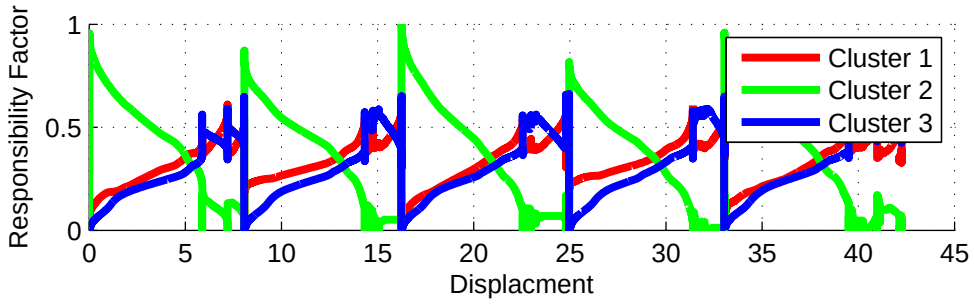
(a) Snapshots from the robot opening bottle *b*4



(b) Cap displacement during the robot's opening

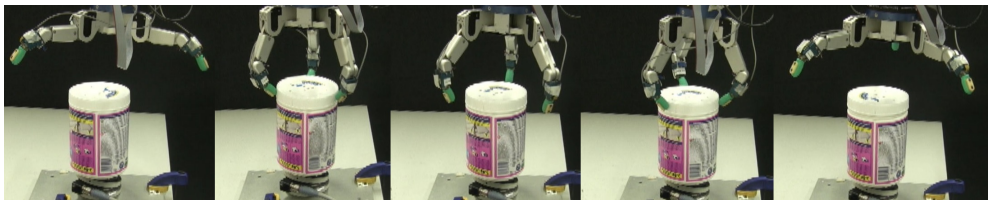


(c) Torque exerted by the robot against cap displacement

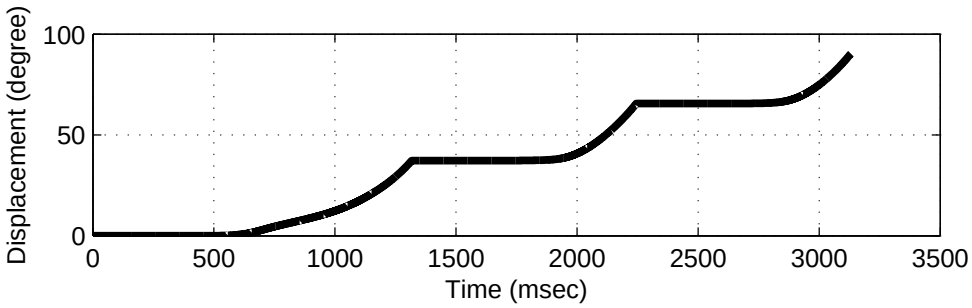


(d) Responsibility factor against cap displacement, for each module

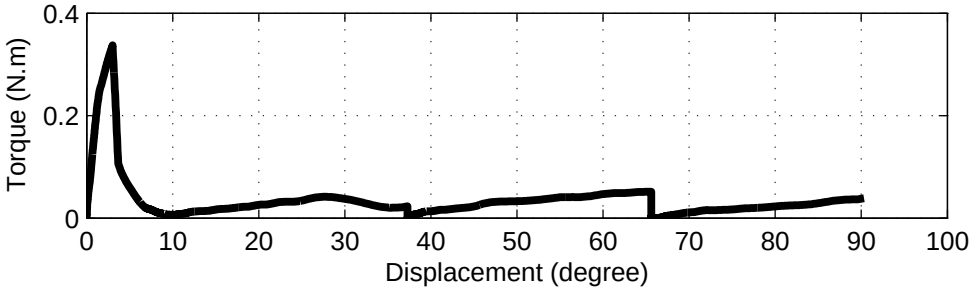
Fig. 15: The robot opens bottle *b*4



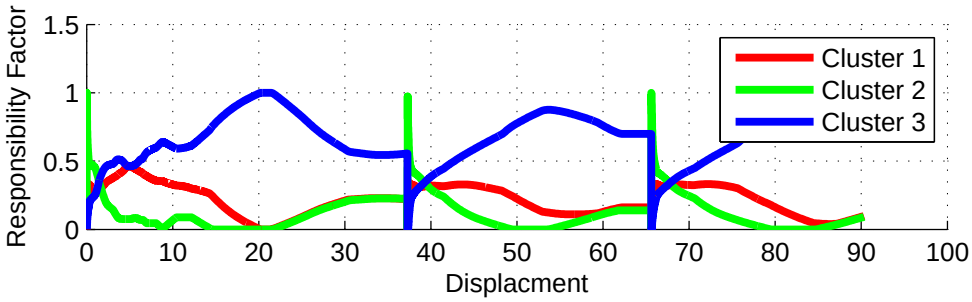
(a) Snapshots for robot opening bottle b5 demonstration



(b) Cap displacement during the robot's opening

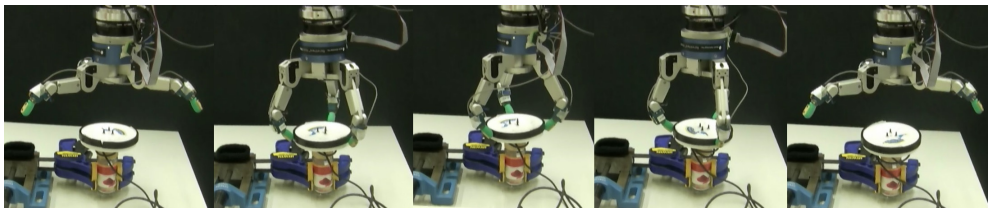


(c) Torque exerted by the robot against cap displacement

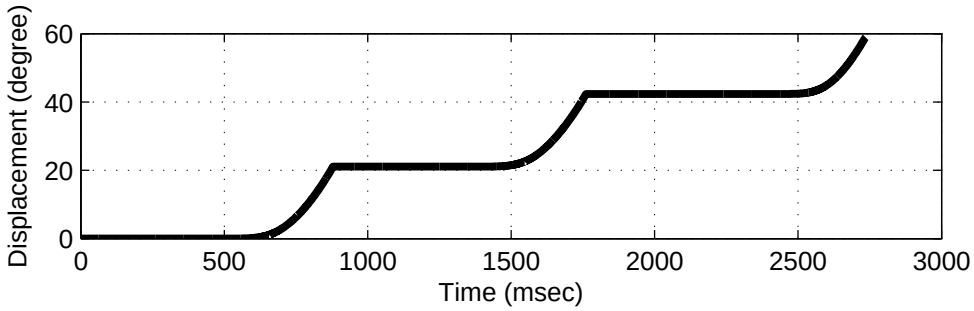


(d) Responsibility factor against cap displacement, for each module

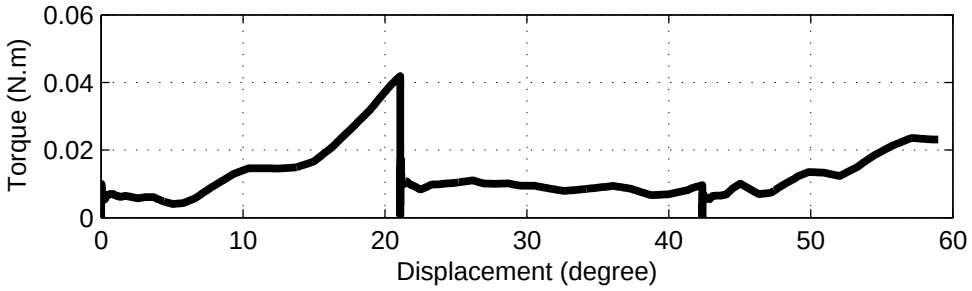
Fig. 16: The robot opens bottle *b5*



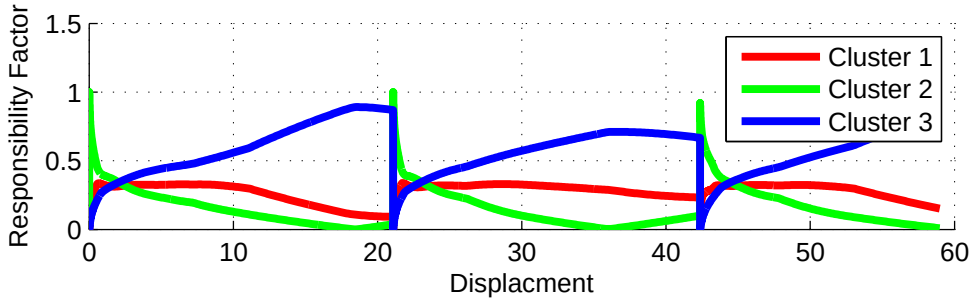
(a) Snapshots for robot opening bottle b6 demonstration



(b) Cap displacement during the robot's opening



(c) Torque exerted by the robot against cap displacement



(d) Responsibility factor against cap displacement, for each module

Fig. 17: The robot opens bottle *b6*

tasks such as impedance control, as measuring human muscle impedance is hard while measuring the impedance of an object is more feasible. Our approach focuses on imitating object movement rather than human movement. For generating natural looking manipulation strategies, however, the object-centric approach does not guarantee good results.

We compute the final motor command by summing the weighted output of each module. This makes an assumption that the state space is continuous. For tasks with discontinuous space, switching between different modules would be more applicable (Narendra et al, 1995; Nakanishi et al, 2013).

An assumption we make during the learning is that the human always uses a single control strategy to handle a particular task context. Based on this assumption, we cluster the training data and modularize the strategy to several modules. The linear combination of these modules can span a large neighbourhood and generate new adaptive strategies for novel task contexts. It is possible that the demonstrations are combinations of more fundamental human control strategies. Decomposing them to these fundamental strategies may allow us to span to a larger space, or even generalize to different tasks. Finding these fundamental strategies will require more demonstrations for different tasks. This will be the next step for extending our approach.

There are many promising directions of further studies extending the work presented here. The first is to apply this approach to other contact tasks and learn a more general human control strategy in handling the instability caused by friction. In our study, we have focussed on the control strategy of unscrewing the cap. We hardly analyzed the effect of changing the cap size and or the positioning of the fingers on the cap, which is revealed in the tactile signature. For the task here, these were not important and did not cluster separately, but for other contexts these could be important. We expect this analysis to advance the study of the task specific grasping strategy (El-Khoury et al, 2013; Dang and Allen, 2014) from the force perspective.

To extend our approach to learn tasks involving multiple steps, one could also integrate this framework with task segmentation techniques, to break down the task into atomic steps and recognize the steps needed, still using a modular approach. However, we could expect this to complicate the point of module integration and require better-informed action selection.

7 Conclusion

In summary, tasks involving multiple phases or different contexts are hard to implement with a single monolithic model. A modular architecture is a practical approach for both learning and controlling these tasks. As manipulation usually involves multi-phase friction and multi-body interaction, learning manipulation tasks with a modular approach can simplify the modeling problem to a significant extent. We have presented here a framework for training a modular model on observed human demonstrations, discovering the strategies used by the humans through a system of cluster analysis, and encoding the results in generative models capable of driving robots. We have demonstrated that we can use this framework to transfer strategies used by a human to a robot, using the task of bottle-cap opening. The demonstration showed not only 'simple' transference from human to robot, but the capacity for generalizing to similar but previously-unobserved contexts, and to adapt sequences of actions in response to the current context.

Acknowledgements This work was funded primarily by the Swiss National Foundation through the National Center of Competence in Research (NCCR) in Robotics. Ravin de Souza was also supported by a doctoral grant (SFRH / BD / 51071 / 2010) from the Portuguese Fundacao para a Ciencia e a Tecnologia and Miao Li was supported by the European Union Seventh Framework ProgrammeP7 / 2007-2013 under grant agreement n^{o} 288533 ROBOHOW.COG. Bidan Huang was also supported by a studentship from the University of Bath. The authors would like to thank Sahar El-Khoury for her valuable comments.

References

Asfour T, Azad P, Gyarfas F, Dillmann R (2008) Imitation learning of dual-arm manipulation tasks in humanoid robots. International Journal of Humanoid Robotics 5(02):183–202

Athans M, Castanon D, Dunn KP, Greene C, Lee W, Sandell Jr N, Willsky AS (1977) The stochastic control of the f-8c aircraft using a multiple model adaptive control (MMAC) method–Part I: Equilibrium flight. Automatic Control, IEEE Transactions on 22(5):768–780

Bernardino A, Henriques M, Hendrich N, Zhang J (2013) Precision grasp synergies for dexterous robotic hands. In: Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on, IEEE, pp 62–67

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD Workshop, Seattle, WA, vol 10, pp 359–370

Bryson JJ (2000) Cross-paradigm analysis of autonomous agent architecture. Journal of Experimental and Theoretical Artificial Intelligence 12(2):165–190

Bryson JJ, Stein LA (2001) Modularity and design in reactive intelligence. In: Proceedings of the 17^{th} International Joint Conference on Artificial Intelligence, Morgan Kaufmann, Seattle, pp 1115–1120

Buchli J, Stulp F, Theodorou E, Schaal S (2011) Learning variable impedance control. The International Journal of Robotics Research 30(7):820–833

Calinon S, Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: Proceedings of the ACM/IEEE international conference on Human-robot interaction, ACM, pp 255–262

Calinon S, Guenter F, Billard A (2007) On learning, representing, and generalizing a task in a humanoid robot. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37(2):286–298

Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. arXiv preprint cs/9603104

Dang H, Allen PK (2014) Semantic grasping: planning task-specific stable robotic grasps. Autonomous Robots pp 1–16

Demiris Y, Khadhouri B (2006) Hierarchical attentive multiple models for execution and recognition of actions. Robotics and autonomous systems 54(5):361–369

Dillmann R (2004) Teaching and learning of robot tasks via observation of human performance. Robotics and Autonomous Systems 47(2):109–116

Do M, Asfour T, Dillmann R (2011) Towards a unifying grasp representation for imitation learning on humanoid robots. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, pp 482–488

El-Khoury S, Li M, Billard A (2013) On the generation of a variety of grasps. Robotics and Autonomous Systems 61(12):1335–1349

Fekri S, Athans M, Pascoal A (2007) Robust multiple model adaptive control (RMMAC): A case study. International Journal of Adaptive Control and Signal Processing 21(1):1–30

Fischer M, van der Smagt P, Hirzinger G (1998) Learning techniques in a dataglove based telemanipulation system for the DLR hand. In: Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, IEEE, vol 2, pp 1603–1608

Flanagan JR, Bowman MC, Johansson RS (2006) Control strategies in object manipulation tasks. Current opinion in neurobiology 16(6):650–659

Gustafsson E (2013) Investigation of friction between plastic parts. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden

Haruno M, Wolpert DM, Kawato M (2001) Mosaic model for sensorimotor learning and control. Neural computation 13(10):2201–2220

Howard M, Mitrovic D, Vijayakumar S (2010) Transferring impedance control strategies between heterogeneous systems via apprenticeship learning. In: Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, IEEE, pp 98–105

Huang B, Bryson J, Inamura T (2013a) Learning Motion Primitives of Object Manipulation Using Mimesis Model. In: Proceedings of 2013 IEEE International Conference on Robotics and Biomimetics. ROBIO

Huang B, El-Khoury S, Li M, Bryson JJ, Billard A (2013b) Learning a real time grasping strategy. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, pp 593–600

Hueser M, Baier T, Zhang J (2006) Learning of demonstrated grasping skills by stereoscopic tracking of human head configuration. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, IEEE, pp 2795–2800

Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural computation 3(1):79–87

Jain A, Kemp CC (2013) Improving robot manipulation with data-driven object-centric models of everyday forces. Autonomous Robots 35(2-3):143–159

Johnson M, Demiris Y (2005) Hierarchies of coupled inverse and forward models for abstraction in robot action planning, recognition and imitation. In: Proceedings of the AISB 2005 Symposium on Imitation in Animals and Artifacts, Citeseer, pp 69–76

Khalil W, Dombre E (2004) Modeling, identification and control of robots. Butterworth-Heinemann

Kondo M, Ueda J, Ogasawara T (2008) Recognition of in-hand manipulation using contact state transition for multifingered robot hand control. Robotics and Autonomous Systems 56(1):66–81

Korkinof D, Demiris Y (2013) Online quantum mixture regression for trajectory learning by demonstration. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, pp 3222–3229

Kortenkamp D, Bonasso RP, Murphy R (eds) (1998) Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems. MIT Press, Cambridge, MA

Kronander K, Billard A (2012) Online learning of varying stiffness through physical human-robot interaction. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, Ieee, pp 1842–1849

Kuipers M, Ioannou P (2010) Multiple model adaptive control with mixing. Automatic Control, IEEE Transactions on 55(8):1822–1836

Kulić D, Takano W, Nakamura Y (2008) Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains. The International Journal of Robotics Research 27(7):761–784

Kulic D, Takano W, Nakamura Y (2009) Online segmentation and clustering from continuous observation of whole body motions. Robotics, IEEE Transactions on 25(5):1158–1166

Kulić D, Ott C, Lee D, Ishikawa J, Nakamura Y (2012) Incremental learning of full body motion primitives and their sequencing through human motion observation. The International Journal of Robotics Research 31(3):330–

345

Li M, Yin H, Tahara K, Billard A (2014) Learning object-level impedance control for robust grasping and dexterous manipulation. In: Proceedings of International Conference on Robotics and Automation (ICRA), 2014., (accepted)

Nakanishi J, Radulescu A, Vijayakumar S (2013) Spatio-temporal optimization of multi-phase movements: Dealing with contacts and switching dynamics. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, pp 5100–5107

Narendra KS, Balakrishnan J (1997) Adaptive control using multiple models. Automatic Control, IEEE Transactions on 42(2):171–187

Narendra KS, Balakrishnan J, Ciliz MK (1995) Adaptation and learning using multiple models, switching, and tuning. Control Systems, IEEE 15(3):37–51

Nehaniv CL, Dautenhahn K (2002) The correspondence problem. In: Dautenhahn K, Nehaniv CL (eds) Imitation in animals and artifacts, MIT Press, chap 2, pp 41–62

Okamura AM, Smaby N, Cutkosky MR (2000) An overview of dexterous manipulation. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, IEEE, vol 1, pp 255–262

Pais AL, Billard A (2014) Encoding bi-manual coordination patterns from human demonstrations. In: Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, ACM, pp 264–265

Pais L, Umezawa K, Nakamura Y, Billard A (2013) Learning robot skills through motion segmentation and constraints extraction. In: HRI Workshop on Collaborative Manipulation

Pastor P, Kalakrishnan M, Chitta S, Theodorou E, Schaal S (2011) Skill learning and task outcome prediction for manipulation. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, pp 3828–3834

Petkos G, Toussaint M, Vijayakumar S (2006) Learning multiple models of non-linear dynamics for control under varying contexts. In: Artificial Neural Networks–ICANN 2006, Springer, pp 898–907

Romano JM, Hsiao K, Niemeyer G, Chitta S, Kuchenbecker KJ (2011) Human-inspired robotic grasp control with tactile sensing. Robotics, IEEE Transactions on 27(6):1067–1079

Sauser E, Argall B, Metta G, Billard A (2011) Iterative learning of grasp adaptation through human corrections. Robotics and Autonomous Systems

de Souza R, El Khoury S, Santos-Victor J, Billard A (2014) Towards comprehensive capture of human grasping and manipulation skills. In: 13th International Symposium on 3D Analysis of Human Movement

Sugimoto N, Morimoto J, Hyon SH, Kawato M (2012) The eMOSAIC model for humanoid robot control. Neural Networks 29:8–19

Tribology-abccom (2014) Coefficient of friction, rolling resistance, air resistance, aerodynamics. URL `http://www.tribology-abc.com/abc/cof.htm`, accessed: 2014-08-09

Willett P (1988) Recent trends in hierarchic document clustering: a critical review. Information Processing & Management 24(5):577–597

Wimböck T, Ott C, Albu-Schäffer A, Hirzinger G (2012) Comparison of object-level grasp controllers for dynamic dexterous manipulation. The International Journal of Robotics Research 31(1):3–23

Wolpert DM, Kawato M (1998) Multiple paired forward and inverse models for motor control. Neural Networks 11(7):1317–1329